🎮 Atividades Incrementais - Primeiros Passos com Pygame

Progressão Pedagógica

Cada atividade adiciona **UM conceito novo** ao código anterior. O aluno vai construindo conhecimento de forma gradual.

✅ Atividade 0: Janela Azul (COMPLETA)

Conceitos aprendidos:

- Inicializar Pygame
- Criar janela
- Game loop básico
- Eventos (fechar janela)
- Atualizar tela

Código base:

```
import pygame
import sys

pygame.init()
screen = pygame.display.set_mode((400, 300))
pygame.display.set_caption("Teste Pygame")

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    screen.fill((0, 100, 200))
    pygame.display.flip()

pygame.quit()
sys.exit()
```

🎯 Atividade 1: Controlar FPS

Objetivo: Aprender sobre controle de framerate e por que é importante.

Novo conceito: `pygame.time.Clock()`

Desafio:

> "A tela está atualizando rápido demais! Vamos controlar para 60 FPS."

O que adicionar:

```
import pygame
import sys

pygame.init()
screen = pygame.display.set_mode((400, 300))
pygame.display.set_caption("Controlando FPS")

# NOVO: Criar objeto Clock para controlar FPS
clock = pygame.time.Clock()

running = True
while running:
    # NOVO: Limitar a 60 FPS (60 frames por segundo)
    clock.tick(60)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    screen.fill((0, 100, 200))
    pygame.display.flip()

pygame.quit()
sys.exit()
```

Explicação para os alunos:

- `clock.tick(60)` pausa o loop para manter 60 FPS
- Sem isso, o jogo roda na velocidade máxima do computador (pode ser 1000+ FPS)
- 60 FPS é suave e consistente entre computadores diferentes

🍫 **KitKat Challenge:** Primeiro a fazer funcionar!

Tempo estimado: 5 minutos

🎯 Atividade 2: Mudando Cores com Teclado

Objetivo: Detectar input do teclado e modificar a tela.

Novo conceito: `event.type == pygame.KEYDOWN` e `event.key`

Desafio:

> "Faça a cor da tela mudar quando apertar teclas: R=vermelho, G=verde, B=azul"

O que adicionar:

```
import pygame
import sys

pygame.init()
screen = pygame.display.set_mode((400, 300))
pygame.display.set_caption("Mudando Cores")
clock = pygame.time.Clock()

# NOVO: Variável para armazenar a cor atual
cor_fundo = (0, 100, 200)  # Começa azul

running = True
while running:
    clock.tick(60)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

        # NOVO: Detectar teclas pressionadas
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_r:  # Tecla R
                cor_fundo = (200, 0, 0)  # Vermelho
            elif event.key == pygame.K_g:  # Tecla G
                cor_fundo = (0, 200, 0)  # Verde
            elif event.key == pygame.K_b:  # Tecla B
                cor_fundo = (0, 0, 200)  # Azul

    # NOVO: Usar a variável cor_fundo
    screen.fill(cor_fundo)
```

```
    pygame.display.flip()

pygame.quit()
sys.exit()
```

Explicação:

- `pygame.KEYDOWN` detecta quando uma tecla é pressionada
- `event.key` diz QUAL tecla foi pressionada
- Tuplas RGB: (vermelho, verde, azul) de 0 a 255

Desafio extra: Adicione mais teclas (Y=amarelo, W=branco, P=preto)

Tempo estimado: 10 minutos

🎯 Atividade 3: Desenhando um Quadrado

Objetivo: Desenhar formas geométricas na tela.

Novo conceito: `pygame.draw.rect()`

Desafio:

> "Desenhe um quadrado vermelho no centro da tela"

O que adicionar:

```
import pygame
import sys

pygame.init()
screen = pygame.display.set_mode((400, 300))
pygame.display.set_caption("Desenhando Formas")
clock = pygame.time.Clock()

running = True
while running:
    clock.tick(60)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Limpa a tela
```

```
    screen.fill((50, 50, 50))  # Cinza escuro

    # NOVO: Desenhar um retângulo vermelho
    # pygame.draw.rect(tela, cor, (x, y, largura, altura))
    pygame.draw.rect(screen, (255, 0, 0), (175, 125, 50, 50))

    pygame.display.flip()

pygame.quit()
sys.exit()
```

Explicação:

- `pygame.draw.rect()` desenha um retângulo
- Parâmetros: (tela, cor, retângulo)
- Retângulo: (x, y, largura, altura)
- (0, 0) é o CANTO SUPERIOR ESQUERDO da tela

Coordenadas:

```
(0,0) ------> X (aumenta para direita)
  |
  |
  v
  Y (aumenta para baixo)
```

Desafio extra: Desenhe 3 quadrados em posições diferentes

Tempo estimado: 10 minutos

🎯 Atividade 4: Movendo o Quadrado com Setas

Objetivo: Mover um objeto com input do teclado.

Novo conceito: `pygame.key.get_pressed()` e variáveis de posição

Desafio:

> "Faça o quadrado se mover com as setas do teclado"

O que adicionar:

```python
import pygame
import sys

pygame.init()
screen = pygame.display.set_mode((400, 300))
pygame.display.set_caption("Movendo Quadrado")
clock = pygame.time.Clock()

# NOVO: Variáveis para posição do quadrado
quadrado_x = 175
quadrado_y = 125
velocidade = 5  # Pixels por frame

running = True
while running:
    clock.tick(60)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # NOVO: Verifica quais teclas estão pressionadas AGORA
    teclas = pygame.key.get_pressed()

    # NOVO: Move o quadrado baseado nas teclas
    if teclas[pygame.K_LEFT]:
        quadrado_x -= velocidade
    if teclas[pygame.K_RIGHT]:
        quadrado_x += velocidade
    if teclas[pygame.K_UP]:
        quadrado_y -= velocidade
    if teclas[pygame.K_DOWN]:
        quadrado_y += velocidade

    # Limpa a tela
    screen.fill((50, 50, 50))

    # NOVO: Desenha o quadrado na posição atualizada
    pygame.draw.rect(screen, (255, 0, 0), (quadrado_x, quadrado_y, 50, 50))

    pygame.display.flip()

pygame.quit()
sys.exit()
```

Explicação:

- `get_pressed()` retorna o estado ATUAL de TODAS as teclas
- Diferente de `KEYDOWN` (evento único), `get_pressed()` verifica continuamente
- Atualizamos as variáveis de posição a cada frame

Diferença importante:

- `KEYDOWN` : Detecta quando aperta (1 evento)
- `get_pressed()` : Detecta enquanto está apertada (contínuo)

Desafio extra: Faça o quadrado mudar de cor quando se move

Tempo estimado: 15 minutos

🎯 Atividade 5: Limitando Movimento (Colisão com Bordas)

Objetivo: Impedir que o quadrado saia da tela.

Novo conceito: Condicionais para limitar movimento

Desafio:

> "Não deixe o quadrado sair da tela!"

O que adicionar:

```
import pygame
import sys

pygame.init()
screen = pygame.display.set_mode((400, 300))
pygame.display.set_caption("Colisão com Bordas")
clock = pygame.time.Clock()

quadrado_x = 175
quadrado_y = 125
tamanho = 50  # NOVO: Guardar o tamanho
velocidade = 5

running = True
while running:
```

```
    clock.tick(60)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    teclas = pygame.key.get_pressed()

    # Move o quadrado
    if teclas[pygame.K_LEFT]:
        quadrado_x -= velocidade
    if teclas[pygame.K_RIGHT]:
        quadrado_x += velocidade
    if teclas[pygame.K_UP]:
        quadrado_y -= velocidade
    if teclas[pygame.K_DOWN]:
        quadrado_y += velocidade

    # NOVO: Limitar posição dentro da tela
    # Não pode ser menor que 0
    if quadrado_x < 0:
        quadrado_x = 0
    # Não pode ser maior que largura da tela - tamanho do quadrado
    if quadrado_x > 400 - tamanho:
        quadrado_x = 400 - tamanho
    # Mesma lógica para Y
    if quadrado_y < 0:
        quadrado_y = 0
    if quadrado_y > 300 - tamanho:
        quadrado_y = 300 - tamanho

    screen.fill((50, 50, 50))
    pygame.draw.rect(screen, (255, 0, 0), (quadrado_x, quadrado_y, tamanho,
tamanho))
    pygame.display.flip()

pygame.quit()
sys.exit()
```

Explicação:

- Verificamos se a posição está dentro dos limites da tela
- Largura da tela = 400, altura = 300
- Se ultrapassar, "empurramos" de volta para o limite

Diagrama:

```
0 ≤ x ≤ (largura_tela - tamanho_quadrado)
0 ≤ y ≤ (altura_tela - tamanho_quadrado)
```

Tempo estimado: 10 minutos

🎯 Atividade 6: Adicionando um Círculo Estático

Objetivo: Desenhar múltiplos objetos e aprender sobre círculos.

Novo conceito: `pygame.draw.circle()`

Desafio:

> "Adicione um círculo amarelo estático na tela"

O que adicionar:

```
import pygame
import sys

pygame.init()
screen = pygame.display.set_mode((400, 300))
pygame.display.set_caption("Quadrado e Círculo")
clock = pygame.time.Clock()

# Quadrado
quadrado_x = 175
quadrado_y = 125
tamanho = 50
velocidade = 5

# NOVO: Círculo (uma "moeda")
circulo_x = 100
circulo_y = 100
circulo_raio = 20

running = True
while running:
    clock.tick(60)

    for event in pygame.event.get():
```

```
        if event.type == pygame.QUIT:
            running = False

    teclas = pygame.key.get_pressed()

    if teclas[pygame.K_LEFT]:
        quadrado_x -= velocidade
    if teclas[pygame.K_RIGHT]:
        quadrado_x += velocidade
    if teclas[pygame.K_UP]:
        quadrado_y -= velocidade
    if teclas[pygame.K_DOWN]:
        quadrado_y += velocidade

    # Limitar bordas
    if quadrado_x < 0:
        quadrado_x = 0
    if quadrado_x > 400 - tamanho:
        quadrado_x = 400 - tamanho
    if quadrado_y < 0:
        quadrado_y = 0
    if quadrado_y > 300 - tamanho:
        quadrado_y = 300 - tamanho

    # Desenhar tudo
    screen.fill((50, 50, 50))

    # NOVO: Desenhar círculo
    # pygame.draw.circle(tela, cor, (centro_x, centro_y), raio)
    pygame.draw.circle(screen, (255, 255, 0), (circulo_x, circulo_y),
circulo_raio)

    # Desenhar quadrado (por cima)
    pygame.draw.rect(screen, (255, 0, 0), (quadrado_x, quadrado_y, tamanho,
tamanho))

    pygame.display.flip()

pygame.quit()
sys.exit()
```

Explicação:

- `pygame.draw.circle()` usa o CENTRO do círculo (diferente do rect)
- Parâmetros: (tela, cor, (x_centro, y_centro), raio)

- A ordem de desenho importa: o que for desenhado depois fica por cima

Tempo estimado: 10 minutos

🎯 Atividade 7: Detectando Colisão

Objetivo: Fazer algo acontecer quando quadrado toca o círculo.

Novo conceito: `pygame.Rect` e `colliderect()`

Desafio:

> "Quando o quadrado encostar no círculo, faça o círculo desaparecer"

O que adicionar:

```
import pygame
import sys

pygame.init()
screen = pygame.display.set_mode((400, 300))
pygame.display.set_caption("Coletando Moeda")
clock = pygame.time.Clock()

# Quadrado (player)
quadrado_x = 175
quadrado_y = 125
tamanho = 50
velocidade = 5

# Círculo (moeda)
circulo_x = 100
circulo_y = 100
circulo_raio = 20
moeda_coletada = False  # NOVO: Flag para saber se coletou

running = True
while running:
    clock.tick(60)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
```

```
    teclas = pygame.key.get_pressed()

    if teclas[pygame.K_LEFT]:
        quadrado_x -= velocidade
    if teclas[pygame.K_RIGHT]:
        quadrado_x += velocidade
    if teclas[pygame.K_UP]:
        quadrado_y -= velocidade
    if teclas[pygame.K_DOWN]:
        quadrado_y += velocidade

    # Limitar bordas
    if quadrado_x < 0:
        quadrado_x = 0
    if quadrado_x > 400 - tamanho:
        quadrado_x = 400 - tamanho
    if quadrado_y < 0:
        quadrado_y = 0
    if quadrado_y > 300 - tamanho:
        quadrado_y = 300 - tamanho

    # NOVO: Criar retângulos para colisão
    # Retângulo do quadrado
    rect_quadrado = pygame.Rect(quadrado_x, quadrado_y, tamanho, tamanho)
    # Retângulo ao redor do círculo (aproximação)
    rect_circulo = pygame.Rect(circulo_x - circulo_raio,
                               circulo_y - circulo_raio,
                               circulo_raio * 2,
                               circulo_raio * 2)

    # NOVO: Verificar colisão
    if rect_quadrado.colliderect(rect_circulo) and not moeda_coletada:
        moeda_coletada = True  # Marca como coletada
        print("Moeda coletada!")  # Mensagem no terminal

    # Desenhar tudo
    screen.fill((50, 50, 50))

    # NOVO: Só desenha o círculo se NÃO foi coletado
    if not moeda_coletada:
        pygame.draw.circle(screen, (255, 255, 0), (circulo_x, circulo_y),
circulo_raio)

    pygame.draw.rect(screen, (255, 0, 0), (quadrado_x, quadrado_y, tamanho,
tamanho))
```

```
    pygame.display.flip()

pygame.quit()
sys.exit()
```

Explicação:

- `pygame.Rect` cria um retângulo invisível para detecção de colisão
- `colliderect()` retorna `True` se dois retângulos se sobrepõem
- Usamos uma flag `moeda_coletada` para controlar o estado

Nota: Colisão com círculo é aproximada (usamos um quadrado). Mais tarde aprenderão colisão circular precisa.

Desafio extra: Adicione 3 moedas e conte quantas foram coletadas

Tempo estimado: 15 minutos

🎯 Atividade 8: Adicionando Pontuação na Tela

Objetivo: Renderizar texto na tela (HUD básico).

Novo conceito: `pygame.font.Font()` e `.render()`

Desafio:

> "Mostre quantas moedas foram coletadas no canto da tela"

O que adicionar:

```
import pygame
import sys

pygame.init()
screen = pygame.display.set_mode((400, 300))
pygame.display.set_caption("Sistema de Pontos")
clock = pygame.time.Clock()

# NOVO: Criar fonte para texto
fonte = pygame.font.Font(None, 36)  # None = fonte padrão, 36 = tamanho

# Quadrado
quadrado_x = 175
quadrado_y = 125
```

```
tamanho = 50
velocidade = 5

# Moedas
moedas = [
    {'x': 100, 'y': 100, 'coletada': False},
    {'x': 300, 'y': 200, 'coletada': False},
    {'x': 200, 'y': 50, 'coletada': False}
]
circulo_raio = 20
pontos = 0  # NOVO: Contador de pontos

running = True
while running:
    clock.tick(60)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    teclas = pygame.key.get_pressed()

    if teclas[pygame.K_LEFT]:
        quadrado_x -= velocidade
    if teclas[pygame.K_RIGHT]:
        quadrado_x += velocidade
    if teclas[pygame.K_UP]:
        quadrado_y -= velocidade
    if teclas[pygame.K_DOWN]:
        quadrado_y += velocidade

    # Limitar bordas
    if quadrado_x < 0:
        quadrado_x = 0
    if quadrado_x > 400 - tamanho:
        quadrado_x = 400 - tamanho
    if quadrado_y < 0:
        quadrado_y = 0
    if quadrado_y > 300 - tamanho:
        quadrado_y = 300 - tamanho

    # Verificar colisão com cada moeda
    rect_quadrado = pygame.Rect(quadrado_x, quadrado_y, tamanho, tamanho)

    for moeda in moedas:
        if not moeda['coletada']:
```

```
            rect_moeda = pygame.Rect(moeda['x'] - circulo_raio,
                                     moeda['y'] - circulo_raio,
                                     circulo_raio * 2,
                                     circulo_raio * 2)

            if rect_quadrado.colliderect(rect_moeda):
                moeda['coletada'] = True
                pontos += 10  # NOVO: Adicionar pontos

    # Desenhar tudo
    screen.fill((50, 50, 50))

    # Desenhar moedas não coletadas
    for moeda in moedas:
        if not moeda['coletada']:
            pygame.draw.circle(screen, (255, 255, 0), (moeda['x'],
moeda['y']), circulo_raio)

    # Desenhar quadrado
    pygame.draw.rect(screen, (255, 0, 0), (quadrado_x, quadrado_y, tamanho,
tamanho))

    # NOVO: Renderizar e desenhar texto de pontuação
    # .render(texto, antialiasing, cor)
    texto_pontos = fonte.render(f'Pontos: {pontos}', True, (255, 255, 255))
    screen.blit(texto_pontos, (10, 10))  # Desenha no canto superior esquerdo

    pygame.display.flip()

pygame.quit()
sys.exit()
```

Explicação:

- `pygame.font.Font(None, tamanho)` cria uma fonte
- `.render(texto, antialiasing, cor)` cria uma superfície com o texto
- `.blit()` desenha a superfície na tela
- f-string `f'Pontos: {pontos}'` para texto dinâmico

Tempo estimado: 15 minutos

📊 Resumo da Progressão

Atividade	Conceito Principal	Tempo	Dificuldade
0	Game loop básico	10min	⭐
1	Clock e FPS	5min	⭐
2	Input de teclado (eventos)	10min	⭐ ⭐
3	Desenhar formas	10min	⭐
4	Movimento contínuo	15min	⭐ ⭐
5	Colisão com bordas	10min	⭐ ⭐
6	Múltiplos objetos	10min	⭐
7	Detecção de colisão	15min	⭐ ⭐ ⭐
8	Renderizar texto (HUD)	15min	⭐ ⭐

Total: ~100 minutos (1h40min) - Perfeito para uma aula de 2h com pausas!

🎯 Atividade Final: Mini-Projeto

Desafio integrador (para casa ou próxima aula):

> "Crie um jogo simples de coletar moedas com:"
>
> - Player que se move com setas
> - 5 moedas espalhadas pela tela
> - Contador de pontos
> - Quando coletar todas, mostrar "VOCÊ VENCEU!"

🍫 **KitKat Grande:** Melhor implementação ganha 3 KitKats!

Critérios:

- Funcionalidade (50%)
- Código limpo e comentado (30%)
- Criatividade (cores, layout) (20%)